Fractalide: A privacy preserving communication platform

Version 0.1.0

Stewart Mackenzie

June 1, 2020

Nearly forty years after the creation of today's Internet we understand many of its shortcomings. Today's Internet has changed the world, the world has moved on and the problem has shifted. The future Internet is information-centric and not host-centric. Enabling an information-centric internet means the user has the content dissemination capabilities of tech giants[1] at negligible costs to them. By disseminating data[2] on terms defined by the user, they no longer need to rely on third party tech giants, which are security holes. Fractalide aims to deliver a platform tailored to preserve the individual's privacy from state actors and sidestep tech giants by granting users tools that promote free speech, make deplatforming impossible, preserve privacy and enable mass dissemination of data from the edges of the network.

[1] Google, Apple, Facebook, Amazon and Microsoft

[2] Copying data such as audio, video, webpages, sofware, ebooks, content in general to whomever requests the data

Contents

1 Introduction

Every action we do online leaves a digital footprint, a trail of data stored by large tech giants. Data that paints a detailed picture of our lives and at a larger scale gives insights into political, cultural and economic trends.

Google knows what happens when two phones come into close proximity to each other at a night club and head off to a motel for the night. Google knows what was on your mind three years down to the second as you searched for an abortion clinic.

Google and other tech giants GAFAM[3] have a significant asymmetric information advantage over you. Should that information fall into the wrong hands it may be used against you.

Tools such as Tor (Dingledine 2010) attempt to wrestle back anonymity by putting data through an onion layered mixer server, though sadly monitoring of Tor exit nodes is the main problem. Hosted VPNs pretend to give you anonymity, which can all too easily be compromised by paying off the VPN for user data.

The only way to reduce the tech giant asymmetric information disadvantage is for the user to retain ownership of their data, for this to happen users must first be able to disseminate their data at scale, independent of tech giants. This starts by removing location data from packets entirely.

[3] Google, Apple, Facebook, Amazon and Microsoft

2 The Problem

The fundamental reason for relying on tech giants is because we are doing a data dissemination over a host-centric communication system.

Data dissemination is the distribution or transmission of data to end users. The current Internet is a host-centric communication system, namely TCP/IP. Essentially a TCP/IP connection is a dumb pipe with an address on each end of the pipe. In one end bits are dumped and out the other end they spill out.

A large-scale content dissemination over a host-centric communication system is an extremely expensive affair. It requires costly hardware, software and networking engineers to maintain millions of concurrent open pipes functioning in a stable fashion. Most small companies cannot afford a large-scale content dissemination over a host-centric communication system and instead prefer to delegate the technical challenges to GAFAM. This is where security and privacy holes are introduced.

Obtaining data in a host-centric communication system means users must go through a host indirection, a data gatekeeper. Where there are gatekeepers you get rent-seekers. It is therefore expected data dissemination monopolies arise as an emergent phenomenon.

2.1 Privacy implications

Content gatekeepers extract large amounts of data from their users. Organisations like Cambridge Analytica use this data bringing an end to free and fair democratic elections by identifying and swinging fence sitter votes. Governments cannot resist tapping into this wealth of information to put the population under surveillance. Gone are the days when a judge's order is required to tap a line, now the entire population is put under mass surveillance and persisted for days in large warehouses for later processing.

Connecting to host addresses to obtain data means we connect to a gate keeper. If the gate keeper is compromised, say by a change of management, a merger and aquisition, the user can be denied access to their content. Increasingly we're seeing the deplatforming of individuals because they do not hold the same political views of the platform owner. Some might argue they're simply choosing not to have these individuals on their platform and if they want others to hear what they have to say then they should create their own platform or find another. This is a correct viewpoint although Google has a long history of delisting competitors.

By moving away from a host-centric networking to an information-centric network (ICN) where the user is in total control of their data. An ICN provides the user with the same data dissemination reach as that of tech giants and allows them to secure their data directly through optional encryption.

3 History of the Internet

Understanding why we have such a centralised infrastructure motivates a decentralised information-centric future. The purpose of this section is to give a brief high level introduction into the phases of the internet and the problems each phase solved. More importantly, this section conveys each phase as a bootstrapper for the next phase.

3.1 Phase 1: The Telephone System - Putting Down the Wires

Alexander Graham Bell[4], along with his company American Telephone and Telegraph Company (AT&T) in 1885, arguably started the Internet, albeit not the Internet we know today. Bell invented the twisted-pair cable, allowing for long distance communication, indeed original AT&T pairs still exist till today in rural areas of America [5].

As networked computers didn't exist, hearing someone's voice was the killer application. Therefore it made sense the best way to amortize costs of pulling pairs between offices and households was to charge for telephone calls. Though telephone calls were purely a side effect of the system. The system on the other hand was doing something entirely different, it was dynamically reconstructing itself in order to make connections between callers and callees. It is important to understand the phone number isn't an end point address instead, it is a program or set of instructions telling phone operators (figure 1) and later step-by-step switches[6] how to build a physical connection between line cards. The phone number was a two dimensional coordinate on a switch board and were intrinsic to the network as AT&T was a monopoly and didn't need an interface to outside networks.

3.1.1 Problems

The underlying infrastructure of the Internet suffered from a few issues (i) intristic addresses which promoted monopolies, (ii) reliablity of connection uptimes.

In order to build a connection in a telephony network all resources need to be known ahead of time. Luckily, this was straightforward for AT&T as it was a monopoly. The nature of intrinsic addresses even caused smaller telcos created in other countries to turn into large monopolies over time.

Telephony networks are structurally unreliable because the more the network scales the probability of failure increases exponentially. As the reliability factor is the product of the failure probability of every component in the connection. So the only two ways to combat this degree of structural unreliability was to improve the quality of each component such that gold

[4] https://en.wikipedia.org/wiki/Alexander_Graham_Bell
[5] https://en.wikipedia.org/wiki/Twisted_pair
[6] https://techchannel.att.com/play-video.cfm/2011/7/22/AT&T-Archives-The-Step-by-Step-Switch

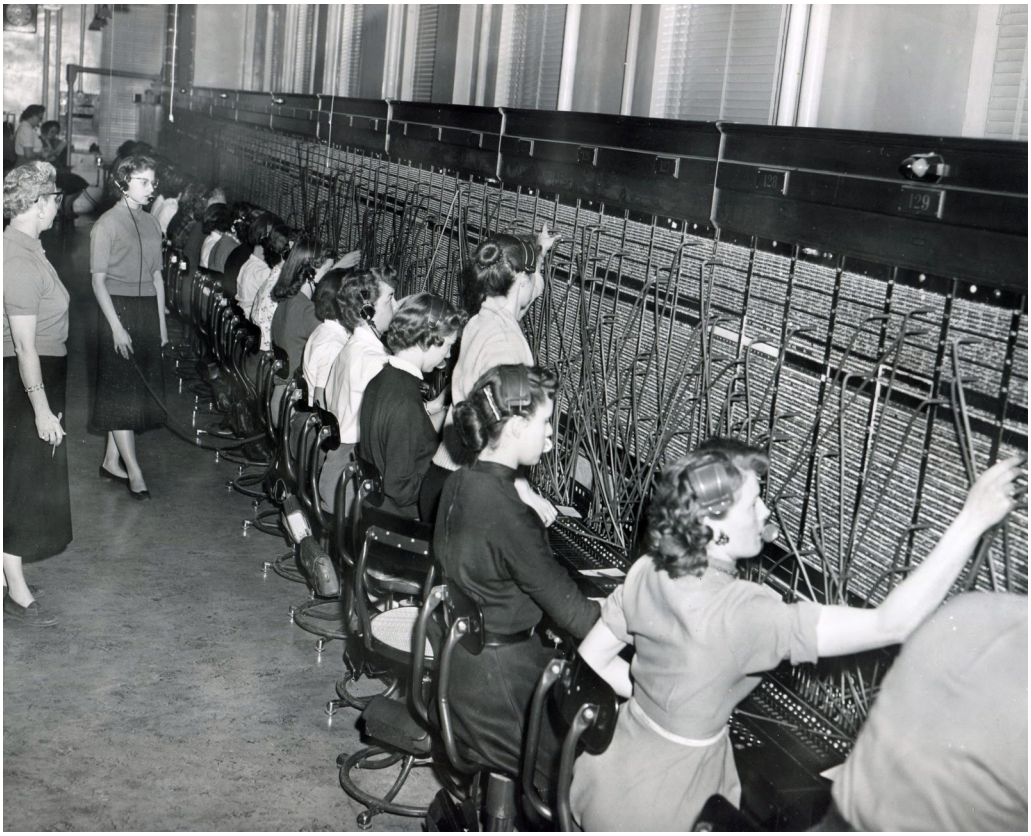


Figure 1: Operators

plated connector with fail over systems were used so that components could last a lifetime. The second way was to ensure hierarchies didn't go greater than 5 deep. Despite such high quality components and conventions in place, AT&T couldn't achieve reliable systems.

The primary problem of this era was laying down wires, which later formed the backbone of the Internet today.

3.2 Phase 2: The Internet - Endpoints or Packet Switching

After the 70ies and 80ies came around, data over telephony became more common in large academic and government institutions in the United States and Europe. Connection setup times took too long to setup. For voice, connection times were sufficient but for data, the

times was impractical, resulting in potential gigabytes of cumulative lost time. Almost the entire networking field focused on reducing connection times, to no avail.

Paul Baran and Donald Davies understood (Baran 1962) that telephony might not be the only model to use for data networks. By this time there was sufficient wires installed in Phase 1 of the Internet, thus allowing Paul and Donald to tackle the problem in different direction. Instead of focusing on physical connection path setup they focused on endpoints. They suggested that the entire network should be physically hooked up. Data should be broken into small packets, with each packet header containing the destination address. Then telling the network to route each packet to the destination address.

In this network the combinatorics are turned upside down, in that, as the system grows in size, the reliablitiy of a successful packet delivery increases exponentially. No longer does one need expensive high quality gold plated components, instead one could throw cheap comodity hardware at the network and the system gets more robust as it scales.

Members of Advanced Research Projects Agency Network (ARPANET) decided to build Paul's network (figure 2).

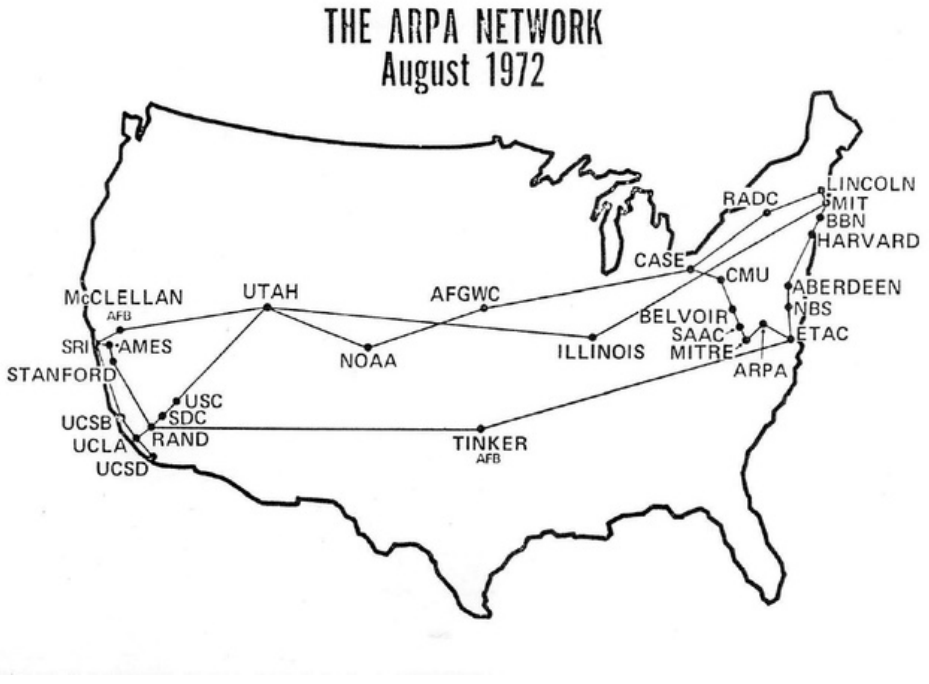


Figure 2: ARPANET

By the time 1793 rolled around packet switching networks had proven themselves. No longer do we care about the topology of the network, in stark contrast to telephony networks. One can dump large amounts of data into packet switching networks and the network will route packets to their destinations. Should a few packets fail then the endpoints reissue requests for the data to fix it up.

Vint Cerf realised one could concatenate different packet switching networks together if one kept the same packet headers between different networks. Therefore each network could be a different type of network be it radio frequency, satellite, wires etc. and packets could find their way through each network to the destination address. Transmission Control Protocol / Internet Protocol (TCP/IP) allowed for the above and was specified in 1974. In 1977 the internetwork demonstation (figure 3) was turned on and we've been running and expanding this TCP/IP for about 42 years since. Humanity's largest and most complex technology system to date.

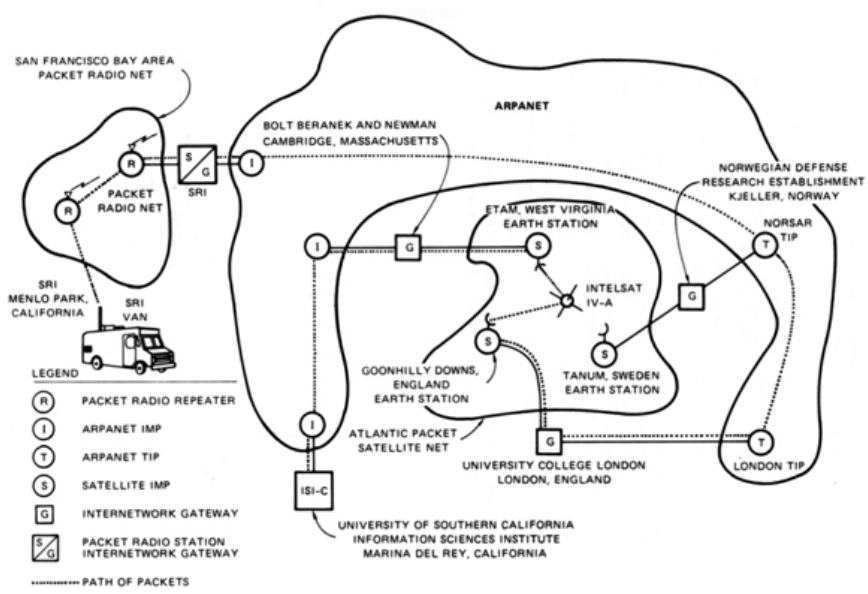


Figure 3: Inter-network Demonstration 1977

3.2.1 Problems Solved

Packet networks have their addresses explicit in the data and no longer implicit on the particular configuration of the wires. This allowed packet switching networks to essentially connect itself whereas telephony networks required humans (or steppers) to rewire path-

ways.

Network reliability increases exponentially the larger the network of comodity hardware becomes, versus telephony networks that become more fragile the larger the network of expensive components gets.

Connection setup time is now no longer a problem as you are already connected to the network and the network will route your packets instead of you needing to setup a telephony pathway to every implicit destination.

Topology of the network has no bearing on packet routing. Whereas telephony networks were hierarchical and strictly constrained to a certain size. Without the size constraint the network easily fails.

Lastly, the network repairs itself. If a packet doesn't go through the endpoints will tell each other about it and fix up the data.

3.2.2 Problems introduced

TCP/IP was created in order to make it easier to connect machines together. TCP/IP was influenced by its predecessor telephony in that TCP/IP was a conversational protocol that involved only two hosts. This is rarely the case when we want to achieve data dissemination. Data dissemination is transmitting data from one location to many locations, it's the main type of communication we do today. In other words, TCP/IP is host-centric networking.

So the model doesn't really fit well for what we want to do. There are other things that don't fit, in that, we are mobile and the TCP/IP inherently doesn't like mobility.

Furthering the mobility point, there are times when we leave network connectivity. This is purely a side effect of the TCP/IP network. One wants a network that doesn't have this binary state of being connected to the Internet or not. If we're near another person's phone we should be automatically be in communication with it.

As TCP/IP requires knowing all end points there needs to be topologically stable end points that are alive at least for the duration of routing times. It's heavy weight maintaining such a large network of globally known addresses.

On the same hand, making a connection with these end points is also expensive, especially when there is a mapping from DNS entries to the IP address you wish to seek out.

Lastly, there is a wealth of communication mediums such as radio frequency that isn't friendly to TCP/IP. Radio frequency is a broadcast medium so and is ubiquitous and cheap. Our communication protocols should be making heavy use of this medium.

TCP/IP solved an old problem of connectivity extremely well. The world has evolved and now faces a new set of problems. Problems TCP/IP is ill suited at solving.

3.3 Phase 3: Data Dissemination - Information Centric Networking

Information-centric Networking (ICN) evolves the current Internet infrastructure to support uniquely named data as a core principle. Data as a result becomes independent of location, storage, application and the very means by which data is transported. This allows for in-network data caching and replication. Security is built into the network as packets are signed and optionally encrypted.

ICN is the umbrella name for a series of different ICN techonolgies all exploring this space. In particular the Fractalide project aims to pursue the Named-data Networking project.

4 Copernica

Fractalide is the overarching project name that consists of a number of subprojects. Copernica is one such subproject and is privacy preserving Information-centric networking protocol designed to operate over UDP and Radio Frequency.

Named after Copernicus, who realised the Earth isn't the center of the Universe just as the IP host isn't the center of networking, instead the Sun or Named Data should be the center of the networking world.

Copernica is licensed under the Mozilla Public License v2 and is implemented in the Rust programming language [7].

4.1 Copernica Principles

Copernica retains three principles that enabled the success of the internet today, and adds a further three principles derived hindsight.

4.1.1 Hourglass Achitecture

A host-centric network might be described by using the hourglass architecture (figure 4[8]). Data flows up through the hourglass and back down over the network. Currently all data flows through the thin waist, that being the IP address. Meaning both sides of the hourglass, applications and app protocols on one side and internet protocols need to interface with the IP address thin waist. The IP address is universal in this model. Copernica shifts the thin waist away from IP Addresses to a Named Chunks, a far more sensible approach when dealing with content, something so central to our lives.

Retaining the hourglass architecture means that the upper and lower parts of the architecture can evolve with sensible constraints such as the name of our data. The upper portion of the hourglass represents constraints of how the data is used, whereas the lower portion of the hourglass represents constraints of how data is in motion (figure 5[9]).

4.1.2 End-to-end principle

The payoffs for adding features to a simple network reduce over time, especially when end hosts need to implement that functionality purely to conform. Meaning the penalities are distributed throughout the network.

[7] https://github.com/fractalide/copernica
[8] http://named-data.net/wp-content/uploads/hourglass.png
[9] https://www.youtube.com/watch?v=5QPTTnRNESg

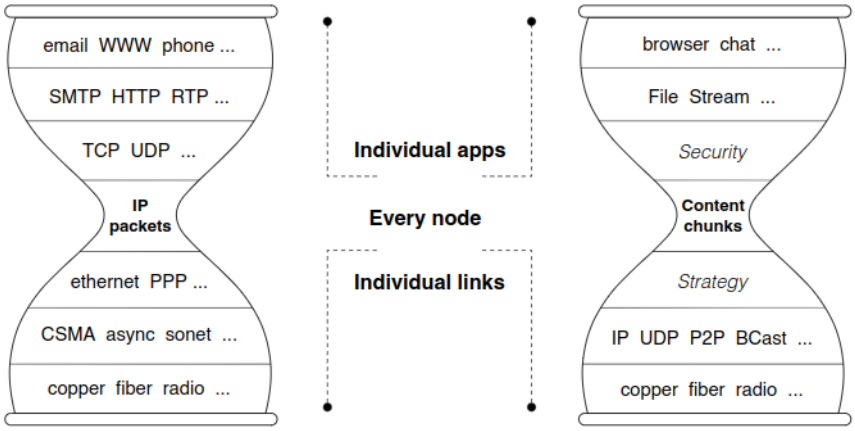


Figure 4: Current Internet Architecture (left) vs Copernica Architecture (right)

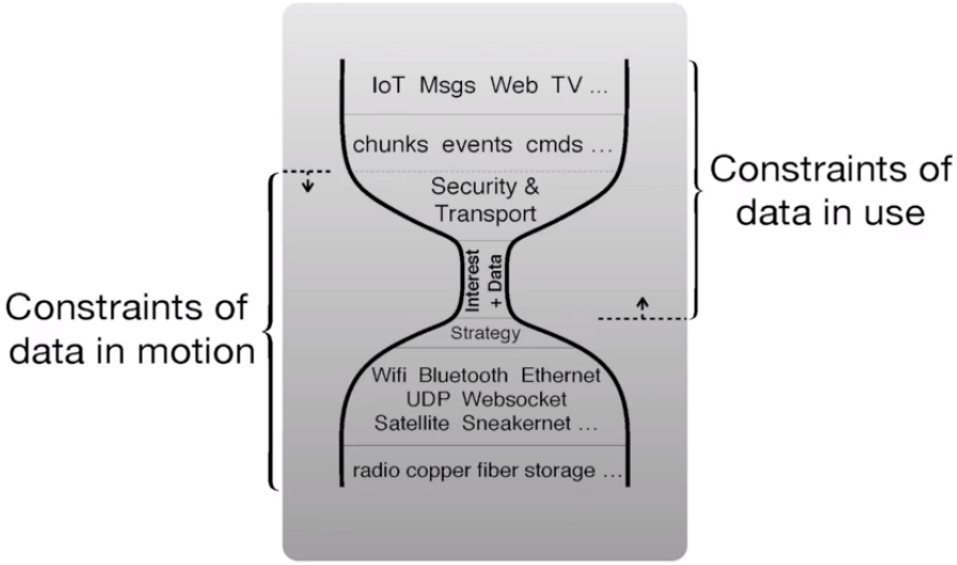


Figure 5: Copernica Hourglass Constraints

The end-to-end principle (Clark 1984) enables robust systems that are resiliant to networking failure. Copernica seeks to retain this feature of keeping the network simple.

4.1.3 Separate routing and forwarding plane

Routing and forwarding planes being separated allowing for the forwarding plane to operate independently of routing, so that routing can incrementally improve over time.

4.1.4 Builtin security from the start

TCP/IP doesn't have security built into it. As a result security has been ad hoc and patched on in a reactionary fashion. Copernica builds strong cryptographic security in at the thin waist of the hourglass so security is ubiquitous throughout the network.

4.1.5 Flow balance

Flow balance ensures that the total flow into a node equals the total flow out of a node. This property is critical for a scalable internet. Copernica ensures that flow balance is adhered to at a hop-by-hop level.

4.1.6 Promote user preferences and competition

A network of components with a standard means of communicating with each other allows for a competitive environment. As long as the API stays the same higher price, less efficient components can be swapped out with more efficient cheaper components. By keeping the standard Copernica packet format ubiquitous throughout the network a common API enables a competitive environment.

4.2 Sparse Distribution Representation

A key design decision is to use Sparse Distributed Representations (SDR) (Hawkins, J. and Ahmad, S. and Purdy, S. and Lavin, A. 2016) in the thin waist instead of hierarchical naming schemes and the associated longest prefix matching used on the forwarding plane.

SDRs in Copernica, serve two purposes; 1) preserving privacy of the data name 2) forwarding data on the forwarding plane.

4.2.1 SDR enabling privacy

Unlike dense representation (ASCII or UTF8 etc.) SDRs are a way of condensing information into a 2048 bitvector at the same time ensuring sparsity. This process has a side effect of creating a unique, succinct fingerprint of the data's name.

For example the name

$$"ceo1q0te4aj3u2llwl4mxuxnjm9skj897hncanvgcnz0gf3x57ap6h7gk4dw8nv :: my-excel-file.xls"$$

when converted into an SDR has this value:

$$[290, 642, 1490]$$

which is an index into a 2048 bitvector.

4.2.2 Forwarding using SDRs

SDRs have a very convenient function. We are able to take a Sparse Distributed Representation Index (SDRI) and perform boolean AND and OR operations to determine if a 2048 bitvector contains this SDRI or not.

This feature means we can now ensure each interface (eth, UDP/IP, Radio Frequency etc) gets its own SDR. As data comes in on interfaces the data's SDRI get recorded accordingly. Other interface SDRs are referenced when determining where to forward the data.

4.2.3 Creating an SDR

When publishing data, the data owner will assign a human readable name to the data. To keep it simple lets just use the publishing identity which is a Bech32 address derived from Ed25519 (Edwards-curve Digital Signature Algorithm). The actual data behind this name will be an encrypted secret key and a cleartext public key. Thus anyone with the id (taken from the above example):

$$"ceo1q0te4aj3u2llwl4mxuxnjm9skj897hncanvgcnz0gf3x57ap6h7gk4dw8nv".$$

can obtain the public key of a user. To obtain the SDRI of this id, we take the id and feed it into a SHA3_512 hashing algorithm.

The output of the SHA3_512 is then scanned for the least frequently occurring character and the positions of this character is recorded.

$$SHA3\_512("ceo1q0te4aj3u2llwl4mxuxnjm9skj897hncanvgcnz0gf3x57ap6h7gk4dw8nv")$$

results in a hash of:

$$768ade3da083187a1028dccea3f7e738c76be4c2ef3fd54bfcfd63f67b34fd58$$
$$8698057a3165b941bbe77355541120c7933efc854ffea0dbb80fcfd7f068a4c$$

This hash has the least occuring hexadecimal character being the number 2, it occurs in 3 positions: position 18, 40 and 93 in the hash. As we want to ensure sparsity, therefore if there are 0 or 1 occurances of a hexadecimal character we will ignore it.

Once we have worked out the particular character and the positions thereof we take this information and plug it into this algorthm:

$$2048BitvectorPosition = positionInHash * 16 + decimal(hexadecimalCharacter)$$

Therefore the hexadecimal character "2" in the positions 18, 40 and 93 can be represented by:

$$[(18, "2"), (40, "2"), (93, "2")]$$

Each array position delimited by a comma, gets its own 2048BitvectorPosition calculation:

$$[(18 * 16 + 2), (40 * 16 + 2), (93 * 16 + 2)]$$

resulting in:

$$[290, 642, 1490]$$

This is only half the story.The function

$$SDRI$$

wraps a 2048BitvectorPosition function call on each least frequently occurring hexadecimal character positions in the SHA3_512 hash.

To create a universally unique data name and associated means to verify the provenance of data we need to repeat the process for the id and then the union of the id + data name:

$$[(SDRI("ceo1q0te...8nv"))$$
$$, (SDRI("ceo1q0te...8nv :: my - excel - file.xls"))]$$

This ensures the receiver of information can query the network using the id to obtain the public key, which is used to verify the signature of the data and optionally decrypt the data.

The above yields our SDRI, a unique anonymous identifier of information:

$$[[290, 642, 1490][17, 481, 593]]$$

4.3 Copernica Architecture

Starting with a description of the packet structure which describes the ”Thin Waist“ we’ll branch out describing how interfaces work and the Request + Response forwarding planes.

4.3.1 Packet structure

Copernica’s packet structure represents a simple Request/Response model.

```
"
struct Sdri {
 id: Vec<u16>,
 name: Vec<u16>,
}

enum Packet {
 RequestData { sdri: Sdri },
 ResponseData { sdri: Sdri, data: Data },
}
"
```

For simplicity sake, other parts of the Thin Waist are omitted, such as the total packet count and the sequence number in the Response packet. The signature is also omitted.

The above clearly demonstrates that a simple SDRI goes out into the network as a Request and a Response containing the same SDRI and associated Data comes back. Given the data is available on the network.

A more sophisticated structure would look like this:

```
"
struct Sdri {
 id: Vec<u16>,
 name: Vec<u16>,
}
```

```
enum Packet {
 RequestData { sdri: Sdri },
 ResponseData { sdri: Sdri, data: Data },
 RequestPipe { sdri: Sdri },
 ResponsePipe { sdri: Sdri, data: Data },
}
"
```

By differentiating between static named data and named pipes we add complexity to the Thin Waist, which enables bidirectional streaming of data and voice-over-copernica type applications.

4.3.2 SDR Privacy implications

Should a state level threat actor intercept a Request packet they would just see:

$$[[290, 642, 1490], [17, 481, 593]]$$

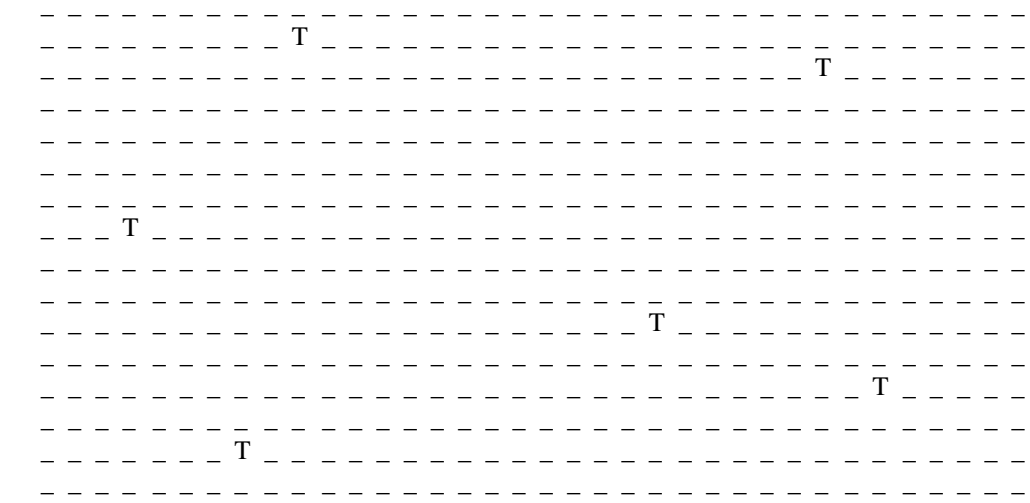
Similary should a state level threat actor intercept a Response packet they would be presented with this:

$$[[[290, 642, 1490], [17, 481, 593]][\begin{array}{c} EnCt2d5f14f76c5e5c3d6d5c6be14487c \\ 509ad5702bc0d5f14f76c5e5c3d6d5c6b \\ e14I + vuDw9yVgG0HbBR7l28RJTVIhIIIB \\ PIQK1x + OOxOZpCrlQFWClEI/9mNC8/LYx \\ vOowGYEmPN3IYlwgoZpk/4ub5YJEbpeYg \\ 8LdkHGUfsLyqedNTSKTFe4 + tvRfc3wi5o \\ roltf32CVkIlQ == IwEmS \end{array}]]$$

Unless one knows the original names, the data names and their associated data payloads are unintelligible. It is through this means that Copernica is able to maintain a privacy preserving architecture such that the data itself is secured along with provenance. Thus packets can flow over insecure channels, indeed through the servers of secret services the world over and nothing much can be done to the integrity of the data nor the cleartext be obtained.

4.3.3 Request & Response SDRIs are inserted into interface SDRs

A SDR is a 2048 bitvector which resembles the below diagram. Each element is initially set to ”false“ denoted as ”_“. When a Request or Response SDRI is inserted into an interface’s SDR it would change the relevant index entries to the value ”true“ denoted as ”T“.



Thus each SDR is capable of holding vast amounts of information.

4.3.4 Interfaces

Copernica might have many interfaces, one could imagine interfaces for websockets, UDP/IP, Radio Frequency, etc. Regardless, each interface needs to have three Sparse Distributed Representation bitvectors associated with it, namely the Pending Request SDR, Forwarding Hint SDR and the Forwarded Request SDR.

Pending Request SDR

The Pending Request Sparse Distributed Representation is used to determine the direction of upstream, meaning, should an inbound Request with SDRI X be present in on Pending Request SDR of that interface, the Request is dropped as the router has already processed this packet. If, however, the Pending Request SDR does not contain SDRI X, the router inserts the SDRI into the Pending Request SDR and then determines where best to forward the Request downstream. An SDRI in a Pending Request SDR is a breadcrumb used to hint on the path a matching Response should return downstream.

Forwarding Hint SDR

The Forwarding Hint Sparse Distributed Representation is used to determine if a request can be satisfied on this inbound interface. Should an SDRI exist in a Forwarded Request SDR the probability this interface being able to satisfy the request is higher than other interfaces.

Forwarded Request SDR

The Forwarded Request Sparse Distributed Representation is used to determine if a request

has been forwarded on another interface (other than the originating interface) so that the router does not to forward the request on the face again, thus violating flow balance.

4.3.5 Forwarding Plane

Copernica routes and forwards by SDRI alone.

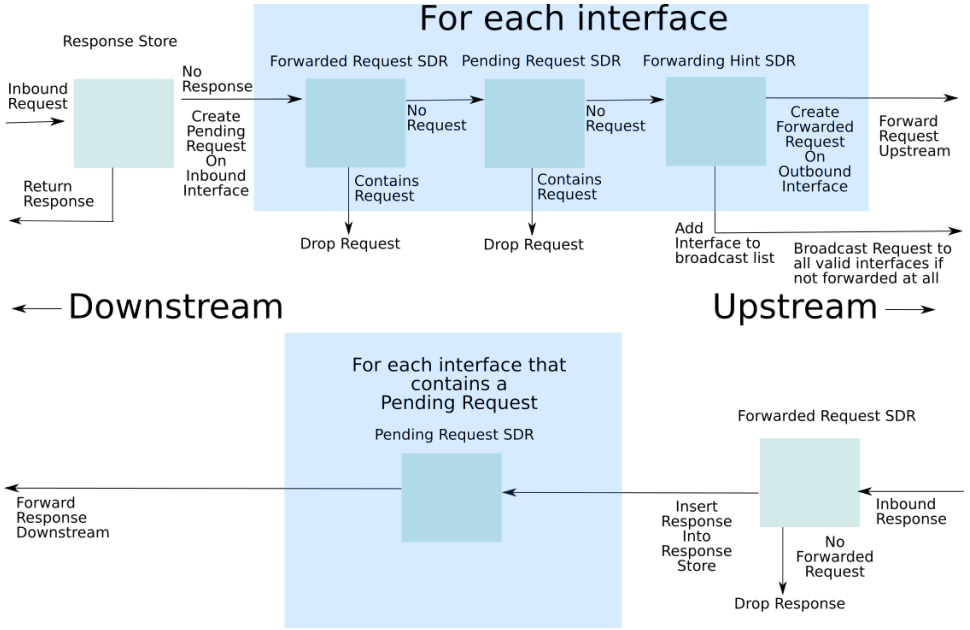


Figure 6: Operational flow in the forwarding plane

Upstream

First we describe the Request Forwarding Plane. A user sets up their CopernicaNet node and peers it with friends CopernicaNet nodes. These connections don't necessarily need to be armoured. We shall call each of these connections a "face".

Our user now seeks data by the name of:

$$\text{“} < friendid > :: movie.mp4''$$

The client library of Copernica will take this name and generate a valid SDRI and construct well formed Request Packet. This packet is then sent to a Copernica Router.

The first thing the router does is examine the Response Store to see if it contains any Responses where the key is exactly the same as the SDRI our user just generated. If there is such a Response, it is immediately returned to our user. Though if the Response Store contains no such data it will then insert the SDRI entry into the inbound face Pending Request SDR. The router will then query the SDRs of each face.

For each face, the router checks to see if there is such an SDRI entry in the Forwarded Request SDR, if there is such an entry, the Request is dropped. As the router has already seen this Request and handled it. If on the other hand there is no SDRI entry in the Forwarded Request SDR then the router will immediately check the outbound face's Pending Request SDR.

If the outbound face's Pending Request SDR contains an SDRI entry exactly the same as our user's generated SDRI then we drop the Request Packet as this means this SDRI has actually come in on this face, so it's pointless sending the Request back to the requestee. Given that the outbound face's Pending Request SDR has no entry then will check to see if the outbound face's Forwarding Hint SDR has an entry.

If the outbound face's Forwarding Hint SDR contains an entry then we immediately forward the request on that face. The Forwarding Hint SDR tells us this particular face has satisfied similar requests before, so the probability is high it can satisfy them again. If on the other hand there is no forwarding hint then we will take this face and add it to a broadcast list. Should we iterate over every face and none have a satisfactory forwarding hint, then we will broadcast this Request on every face in the broadcast list.

Bayes algorithm can be used to optimise the above process.

Downstream

Say the Response exists somewhere out on the CopernicaNet. The Response's SDRI will first be checked at the nodes inbound face Forwarded Request SDR. If no entry exists then we immediately drop the Response. The SDRI not being in the Forwarded Request SDR means the node didn't froward a Request on the face in the first place.

Should there be a SDRI entry in the Forwarded Request SDR then we will insert the Response into the Response Store. Finally we will iterate over every face checking to see if any face has a corresponding SDRI in its Pending Request SDR.

Eventually the Response will return allowing our user to use the data in the Response.

5 Conclusion

Copernica enables anonymous communication. Unlike Tor, Copernica doesn't utilize the onion network to hide users' traffic. Instead Copernica strips out all location data (IP address information) totally and routes via SDRIs. Data is (optionally) encrypted directly and not via armoured pipes. Thus well funded state level threat actors will only be able to intercept irreversible SDRIs and intelligible encrypted data.

Copernica takes a step further in that divorcing location from data, we're able to utilize the network itself to store and forward data. Hence users interested in popular data do not have to traverse the globe to obtain the data. They just need to go to their next door neighbour and copy that data over.

This side effect allows users to step out from under big tech data dissemination monopoly and disseminate data on their own terms. This is the first step in fixing the security holes trusted third parties introduce.

Bibliography

P. Baran. On Distributed Communications Networks. IEEE Transactions on Communications Systems, 1962. http://pages.cs.wisc.edu/~akella/CS740/F08/740-Papers/Bar64.pdf

J.H. Saltzer, D.P. Reed and D.D. Clark. End-to-End Arguments in System Design. M.I.T. Laboratory for Computer Science, 1984. http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf

Karsten Loesing and Steven J. Murdoch and Roger Dingledine. A Case Study on Measuring Statistical Data in the Tor Anonymity Network. Tenerife, Canary Islands, Spain, 2010.

Hawkins, J. and Ahmad, S. and Purdy, S. and Lavin, A. Biological and Machine Intelligence (BAMI). 2016. https://numenta.com/resources/biological-and-machine-intelligence/